[SGI 879]

UNITED STATES PATENT APPLICATION FOR

A SYSTEM AND METHOD FOR MAINTAINING AND RECOVERING DATA CONSISTENCY IN A DATA BASE PAGE

Inventor(s):

Robert G. Mende Jr.

Prepared by:

WAGNER, MURABITO & HAO
Two North Market Street, Third Floor
San Jose, California 95113
(408) 938-9060

A SYSTEM AND METHOD FOR MAINTAINING AND RECOVERING DATA CONSISTENCY IN A DATABASE PAGE

FIELD OF THE INVENTION

The present invention relates to the field of data consistency maintenance and recovery. More particularly, the present invention relates to the maintenance and recovery of valid information in a memory mapped data base utilized by multiple processes operating on a computer system.

BACKGROUND OF THE INVENTION

Electronic systems and circuits have made a significant contribution towards the advancement of modern society and are utilized in a number of applications to achieve advantageous results. Numerous electronic technologies such as digital computers, calculators, audio devices, video equipment, and telephone systems have facilitated increased productivity and reduced costs in analyzing and communicating data, ideas and trends in most areas of business, science, education and entertainment. Electronic systems designed to provide these results are usually arranged in a variety of complicated configurations governed by processing and communication

limitations, including time dependencies and ordering constraints. Typically, electrical systems depend upon consistent or "valid" data to operate properly.

Numerous electrical systems include a variety of processes and frequently different processes attempt to manipulate data at the same time, often resulting in inconsistent ("invalid") data. For example, if a first process in a computer system accesses "old" information that is currently being updated by a second process, the old data may no longer be valid and the first process may produce inaccurate or inappropriate results. In sequential processing and communication systems information is usually divided into units that are transmitted or processed one piece at a time, with one piece of information following another. In some situations it is critical for a first piece of information to follow second piece of information and if the first piece of information is not valid the results are typically unreliable. Maintaining data consistency is particularly important in a computing environment utilizing a database shared by various programs or processes. Processes utilizing a memory mapped data base memory (MDBM) expect that the data they access from a main memory is consistent with the data in a MDBM file and looks the same to any process accessing it. Thus, most computer systems typically require data to be consistent ("correct") with respect to a particular point in time.

Computer system crashes or process crashes typically have an adverse affect on data consistency maintenance. If a system or process crashes in the middle of performing a write transaction the resulting data is typically unreliable and often invalid. Maintaining consistency across process transactions is critical for proper recovery from a process or system crash. If a read or write process (or the system that a write is happening on) crashes anytime during the transaction it is important for the consistency (e.g., of a database) to be maintained so that continuing processes receive valid information.

What is required is a system and method that facilitates data consistency maintenance during a write operation. The system and method should also facilitate recovery from a system or process crash with valid data.

SUMMARY OF THE INVENTION

The present invention is a system and method that facilitates data consistency maintenance between two segments of memory. In one exemplary implementation, the present invention facilitates consistency maintenance during a write operation to a database. The present invention also facilitates recovery from a system or process crash with valid data. A data consistency maintenance and recovery system and method of the present invention utilizes a dual page configuration and locking process to store and track data. A primary page is utilized as the primary data storage location and a mirror page operates as a copy of the primary page except during certain stages of a data manipulation operation (e.g., a write operation). In one embodiment of the present invention, a process can not access a page to perform a read operation if the page is locked and a process can not perform a write operation if the process did not lock the page.

In one embodiment of the present invention, a consistency maintenance locking method and a write tracking method are utilized to facilitate consistency maintenance and recovery from a process or system crash. In one embodiment of the present invention data being manipulated (e.g., changed by a write operation) is stored on a single page. Read operations access information from unlocked primary pages. A write process acquires a genlock of a mirror page and syncs the mirror page to disk. The write process

then performs the update and syncs the mirror page to disk. The write process acquires the genlock on the primary page and syncs it to disk. The write process performs an update on the primary page and syncs it to disk. It then releases the genlock on the primary page and syncs it to disk. It then release the genlock on the mirror page and syncs it to disk.

In one embodiment of the present invention, a primary page is considered consistent if a write operation has not accessed the primary page to begin a write process, otherwise data on a mirror page is considered consistent. In one embodiment of the present invention, a write operation is dropped or continued upon determination that data is inconsistent when recovering from a process or system crash.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1A is a block diagram of a data consistency maintenance and recovery computer system of the present invention.

Figure 1B is a block diagram illustrating one exemplary configuration of pages maintained in a database of the present invention.

Figure 2 is a flow chart of one embodiment of a present invention data consistency maintenance method in which data being modified is included on a single page.

Figure 3 is a flow chart of one embodiment of a present invention data consistency recovery method in which data being modified is included on a single page.

Figure 4A is a block diagram illustrating one exemplary configuration of pages maintained in database by the present invention.

Figure 4B is a flow chart of a data consistency maintenance method for information buckets spread across multiple pages, one embodiment of the present invention.

Figure 5 is a flow chart of a data consistency recovery method of one exemplary implementation of the present invention in which information buckets are spread across multiple pages.

Figure 6 is a block diagram illustrating one exemplary present invention configuration of information stored in multiple index database system.

Figure 7 is a flow chart of a data consistency maintenance method for multiple instances, one embodiment of the present invention.

Figure 8 is a flow chart of a data consistency recovery method for multiple instances, one embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Reference will now be made in detail to the preferred embodiments of the invention, a method and system for maintaining and recovering data consistency in a computer system database, examples of which are illustrated in the accompanying drawings. While the invention will be described in conjunction with the preferred embodiments, it will be understood that they are not intended to limit the invention to these embodiments. On the contrary, the invention is intended to cover alternatives, modifications and equivalents, which may be included within the spirit and scope of the invention as defined by the appended claims. Furthermore, in the following detailed description of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be obvious to one ordinarily skilled in the art that the present invention may be practiced without these specific details. In other instances, well known methods, procedures, components, and circuits have not been described in detail as not to unnecessarily obscure aspects of the current invention.

Some portions of the detailed descriptions which follow are presented in terms of procedures, logic blocks, processing, and other symbolic representations of operations on data bits within a computer memory. These descriptions and representations are the means used by those skilled in the data processing arts to

most effectively convey the substance of their work to others skilled in the art. A procedure, logic block, process, etc., is here, and generally, conceived to be a self-consistent sequence of steps or instructions leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated in a computer system. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussions, it is appreciated that throughout the present invention, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

The present invention is a data consistency maintenance and recovery system and method. In one embodiment of the present invention, a data consistency maintenance and recovery computer system implements a data consistency maintenance method and a data consistency recovery method. In one exemplary embodiment, information is stored in a page based memory mapped, multiple anonymous reader, multiple writer database such as a memory mapped database memory (MDBM). A data consistency maintenance and recovery system and method of the present invention facilitates the reduction of data corruption by process or system crashes that occur during a write procedure. The present invention performs either a roll back or completion of changes to a page being modified during an aborted transaction, depending at which point in the modification process (e.g., write operations) the failure occurred. In one embodiment of the present invention, external transaction logs are not required.

Figure 1A is a block diagram of a data consistency maintenance and recovery computer system 100, one embodiment of the present invention. In general, computer system 100 comprises a central processing unit (CPU) 101, a main memory 102, graphics controller 103, mass storage device 105, keyboard controller 106, network 108, input/output port 109, database 150 and display monitor 110, all of which are coupled to bus 107. CPU 101 handles most of the control and data processing including processes 120 and 130. Main memory 102 provides a convenient method of storing data for quick retrieval by CPU

101. Graphics controller 103 processes image data in pipelined stages. Mass storage device 105 stores data associated with multiple images and applications. Keyboard controller 106 controls keyboard 108, which operates as an input device. Input/output port 109 provides a communication port for a variety of devices. In one exemplary implementation of the present invention, input/output port 109 provides a communication port to a network of devices (not shown). Display monitor 110 displays graphical images. Database 150 stores information arranged in files on a computer readable medium, including records comprising fields for facilitating operations such as searching, sorting, reconfiguring, etc. Bus 107 provides a communication path between components of computer system 100.

In one embodiment of the present invention, database 150 is a memory mapped data base (MDBM) that facilitates multiple anonymous untrusted readers and multiple writers inside an unmanaged shared memory database. In one exemplary implementation of the present invention, a MDBM uses memory mapped files to give memory semantics to a file enabling the data that is accessed from main memory and the data on a file in a database to appear the same to multiple processes accessing the data. In one embodiment of the present invention, for every "primary" page of data in the MDBM a "mirror" page of that data is kept.

Figure 1B is a block diagram illustrating one exemplary configuration of pages maintained in database 150 by one embodiment of the present invention. Each of the primary pages 170 through 173 has a mirror page 181 through 183 respectively. A mirror page is essentially a copy of the primary page and comprises the same data except for certain short periods of time during a write operation when a mirror page is written to first. During the initial stages of a write operation, the mirror page comprises "new" data and a primary page includes "old" data. Each page 170 through 183 includes a genlock that comprises a write counter value 141 through 146 respectively with a low order bit 191 through 196 respectively which serve as a lock bit. The genlock indicates a page is locked or unlocked. In one embodiment of the present invention, a page is locked during operations associated with manipulation of data included in the page (e.g., consistency maintenance write operations). A page is unlocked if data on the page is not being manipulated. If the page is locked (e.g., a genlock's low order bit is set to a logical 1) the page is not available to be accessed by a read operation or accessed by a write operation that did not lock it. If the page is unlocked (e.g., a genlock's low order bit is set to a logical 0) the page is available to be read or write access.

In one embodiment of the present invention, the logical state of the lowest order bit of the write counter value indicates if the page is locked or unlocked. In one exemplary implementation, a logical 1 write counter value

low order bit indicates the page is locked and a logical 0 write counter value low order bit indicates the page is unlocked. In one embodiment of a present invention genlock, when a page is initially accessed in a write operation, the write counter value is incremented and the low order bit changes from a logical 0 value to a logical 1 value indicating the page is locked by a process (e.g. process 120 performing a write operation). In one embodiment of the present invention, write counter values are not incremented during bit wraparound when it's maximum unit (e.g., MAX_UNIT) size is passed. Once the page is locked read operations and other write operations are prevented from accessing the data. When the process has completed the write operations the write counter value is incremented and the low order bit changes from a logical 1 to a logical 0 indicating the page is unlocked. In one embodiment of the present invention, read operations do not require the genlock to be locked since they just read data and do not manipulate it, therefore they do not have the potential to disrupt the consistency or validity of the data.

In one embodiment of the present invention, processes (e.g., process 120 or 130) access a primary or mirror page depending upon the type of operation. In one embodiment of the present invention, read operations access the primary page to obtain data and do not lock a page (e.g., do not increment the write counter value). In one exemplary implementation, write operations first access and modify data in the mirror page and then access and

modify data in the primary page. The write operation does lock a page when it begins a write and unlocks it when it is finished.

Figure 2 is a flow chart of data consistency maintenance method 200, one embodiment of the present invention. Data consistency maintenance method 200 utilizes a genlock to prevent reads from accessing data in the process of being changed by a write operation. The genlock for a mirror page and it's associated primary page are kept in lockstep, except during portions of a write operation. In one embodiment of the present invention, data subject to a write operation is stored in a single primary page and a copy of the data is stored in its associated mirror page. In one embodiment of the present invention, data consistency maintenance method 200 is implemented on a computer system (e.g., data consistency maintenance and recovery computer system 100).

In step 210 a process (e.g. process 120) requests write access to a particular mirror page (e.g., mirror page 181) and attempts to acquire a genlock on that page. The write access request is denied if the genlock of that particular mirror page is locked (e.g., low order bit 194 is a logical 1 value). The write access request is granted if the genlock of that particular mirror page is unlocked (e.g., low order bit 194 is a logical 0 value). In one embodiment of the present invention, the particular mirror page (e.g., mirror page 181) is retrieved from a database (e.g., database 50) and mapped into a local or main

memory (e.g., main memory 102) if not already stored in the local or main memory.

If the write access request is granted and a genlock is acquired, the requesting process (e.g., process 120) locks the genlock (e.g., the write counter value is incremented) and the mirror page is synced to disk in step 220. Locking the genlock provides an indication that the mirror page (e.g., mirror page 181) is being accessed by a process performing a manipulation of the data included in the mirror page (e.g., a write operation). In one embodiment of the present invention, the genlock indication is provided by the least significant bit of the write counter being in a logic level "1" state.

In step 230, the data included in a mirror page is updated and the mirror page is synced to disk. In one embodiment of the present invention, syncing operations are to a backing store or some common point. In one exemplary implementation, syncing operations include a network transaction.

In step 240, data consistency maintenance method 200 acquires the genlock on the primary page and syncs the primary page to disk. In one embodiment of the present invention, the genlock is locked when it is synced back to disk (e.g., incrementing a write counter value).

Data consistency maintenance method 200 performs an update of data on the primary page and syncs the primary page to disk, in step 250.

In step 260, read operations and write operations of other processes (e.g., process 130) are prevented from accessing a locked page. For example, a second process (e.g., process 130) attempting to access a locked page (e.g., primary page 171) checks a genlock indication (e.g., low order bit 191) of the particular page (e.g., page 171) and determines that the page is locked (e.g., the least significant bit of write counter value 141 is in the logic level "1" state). The second process (e.g., process 130) waits to read or write access the particular page (e.g., page 171) until the second process determines the particular page is unlocked (e.g., a logic level "0" state in the least significant bit of write counter value 141). Upon determining the mirror page is unlocked, the second process accesses the particular page to perform an operation (e.g., a read or write operation).

In step 270, data consistency maintenance and recovery method 200 releases the genlock on the primary page and syncs the primary page to disk. In one embodiment of the present invention the genlock is released by incrementing a write counter value associated with the primary page.

In step 280, data consistency maintenance method 200 releases the genlock on the mirror page and syncs the mirror page to disk.

Figure 3 is a flow chart of data consistency recovery method 300, one embodiment of the present invention. Data consistency recovery method 300 facilitates the recovery of consistent data after a process or system crash. The method permits a process that is performing data manipulation (e.g., a write operation) in a database to determine what data is consistent (all writes completed successfully) and what data is not consistent following a process or system crash. In one embodiment of the presentation, data consistency recovery method 300 allows any arbitrary process performing a write operation to either complete or abort the transaction that crashed. In one embodiment of the present invention, data consistency recovery method 300 is implemented in a system (e.g., data consistency maintenance and recovery computer system 100) utilizing a data consistency maintenance method (e.g., data consistency maintenance method 200).

In step 310, the write counter value of a primary page is compared to the write counter value of a mirror page and the comparison is an atomic operation. In one embodiment of the present invention, data consistency recovery method 300 utilizes the comparison to establish if the genlocks of the respective pages are locked or unlocked. In one embodiment of the present invention, data consistency recovery method 300 also utilizes the comparison to determine if the write counter value of a primary page is equal to or less than the write counter value of an associated mirror page.

In step 320, data consistency recovery method 300 determines which pages include valid data. If the write counter values of both the primary page and mirror page are equal and they are both unlocked (e.g., the genlock is a logical 0 value) then the data on both the pages is considered valid and consistent. The information on the primary page is considered consistent if the write counter value of the primary page is less than the write counter value of the mirror page and the primary page is unlocked and the mirror page is locked. The information on the mirror page is consistent if the write counter value of the primary page is equal to the writecounter value of the mirror page and the primary page and the mirror page are locked. The information on the mirror page is consistent if the write counter value of the primary page is greater than the write counter value of the mirror page and the primary page is unlocked and the mirror page is locked. The pages are considered to be in an invalid state if the primary page is locked and the mirror page is unlocked. The pages are also considered to be in an invalid state if the write counter value on both the primary and mirror pages are equal. When an invalid state occures data included in the mirror page is considered the valid information.

In step 330 of data consistency recovery method 300, the consistent page is copied to the inconsistent page and genlock status resolved. In one exemplary implementation of the present invention the genlock status is

resolved by unlocking the genlocks on both the primary page and the mirror page. In one embodiment of the present invention in which the lowest order bit of a write counter value functions as a genlock indication, the writecounter values in a primary page and its associated mirror page are manipulated to equal one another after a recovery and consistency process is performed.

In one embodiment of the present invention, data is stored across multiple pages. In one exemplary implementation of the present invention a database (e.g., a MDBM) allows for storage of relatively large size data. The database breaks the data into "buckets" of various sizes that are stored over multiple pages. The data buckets are "chained" together and referenced by a key (e.g., KEY) comprising an appended bucket identification (ID). In one embodiment of the present invention, the key is a mono atomically increasing unsigned integer. When the data is retrieved the parts or buckets of data are coalesced back together again. In one exemplary implementation it is possible that more than one bucket of data will be on the same database page and the database (e.g., MDBM) assigns them an element location (e.g., an unsigned integer).

Figure 4A is a block diagram illustrating one exemplary configuration of pages maintained in database 450, one embodiment of the present invention. Database 450 comprises primary pages 411 through 413 and their

associated mirror pages 421 through 423 respectively. In one embodiment of the present invention, database 450 is a MDBM database. A data value is broken into buckets identified by a reference to "KEY" and an appended identification (e.g., KEY0, KEY1, KEY2, KEY3 and KEY4). The buckets of the data value referenced by "KEY" are stored over several pages including primary page 411 through 413 and copies stored in mirror pages 421 through 423. Each bucket starts at an ordinal element location (e.g., element location 0, 1, 2, etc.) within a page. In one embodiment of the present invention, an element location corresponds to a memory location or address offset within each page. Primary pages 411 through 413 and mirror pages 421 through 423 include writecounter values 471 through 479, low order bits 481 through 489 and atomic consistency elements 451 through 459. In one embodiment of the present invention, the atomic consistency element values on the primary pages are set to a logical 0 value and do not change. The atomic consistency element values on the mirror page are set to correspond to an element location of a bucket (e.g., KEY[M]) being modified when the mirror page genlock is locked and returned to a logical zero when the mirror page genlock is unlocked.

Figure 4B is a flow chart of data consistency maintenance method 400, one embodiment of the present invention. Data consistency maintenance method 400 is a method for ensuring consistency in bucket chained data across multiple pages of a page based memory mapped, multiple anonymous

reader, multiple writer database (e.g., a MDBM). One embodiment of data consistency maintenance method 400, includes the genlock and consistency method described above. In one exemplary implementation of data consistency maintenance method 400, a process performing a write operation updates relevant mirror pages (e.g., mirror pages comprising data modified by a write operation), then updates the relevant primary pages, and then returns the database to standard unlocked state. In one embodiment of the present invention, consistency state changes happen in an atomic manner. The nomenclature key [M] refers to bucket number M of a data value referenced by the key. In one embodiment of the present invention, data consistency maintenance method 400 is implemented on a computer system (e.g., data consistency maintenance and recovery computer system 100).

In step 410, for each M in 0 through N, where N is the number of buckets that the value associated with a key has been split into, the writing process acquires and locks the genlock of the mirror page for key [M] (e.g., as described above by incrementing an writecounter value), if the page was not previously locked. When locking the genlock, if M is greater than 0, the atomic consistency element for this page is set to the element location on the mirror page for key[M]. Each accessed mirror page is locked and synced to disk if the writer acquired the genlock.

In step 420, for each M in 0 through N, where N is the number of buckets that the value associated with a key has been split into, the writing process updates the mirror page and syncs the page to disk. An optimization of deferring the syncs until after all the pages have been updated allows coalescing the syncs of multiple buckets on the same page.

In step 430, for each M in 0 through N, where N is the number of buckets that the value associated with a key has been split into, the writing process acquires and locks the genlock of the primary page for key [M] if the page was not previously locked. Each page is synced to a backing store (e.g.,a disk) if the writer acquired the genlock.

In step 440, for each M in 0 through N, where N is the number of buckets that the value associated with a key has been split into, the writing process updates the primary page and syncs the page to disk. In one embodiment of the present invention, an optimization of deferring the syncs until after all the pages have been updated allows coalescing the syncs of multiple buckets on the same page.

In step 450, for each M in 0 through N, where N is the number of buckets that the value associated with a key has been split into, the writing process releases (unlocks) the genlock on the primary page and syncs the primary page to disk.

In step 460, for each M in 0 through N, where N is the number of buckets that the value associated with a key has been split into, then the writing process releases the genlock and sets the atomic consistency element to 0 on the mirror page and syncs the mirror page to disk.

Figure 5 is a flow chart of data consistency recovery method 500, one embodiment of the present invention. Data consistency recovery method 500 is a method for recovering valid bucket chained data across multiple pages of a page based memory mapped, multiple anonymous reader, multiple writer database. In one embodiment of the present invention, data consistency recovery method 500 compares the genlocks and count of a mirror page with the appropriate primary page genlock and count to restore consistency. In one exemplary implementation, data consistency recovery method 500 allows for either roll back or completion of the changes to pages in an atomic manner. Whether the data is rolled back after an aborted transaction depends upon the point in the modification operation where the failure occurred. In one embodiment of the present invention, data consistency maintenance method 400 allows a database (e.g., a MDBM) to reduce corruption of data (e.g., data stored in multiple elements in a MDBM) associated with a process or system crash that occures during a write process. In one embodiment of the present invention, data consistency recovery method 500 is implemented in a system (e.g., data consistency maintenance and recovery computer system 100)

utilizing a data consistency maintenance method (e.g., a flow chart of data consistency maintenance method 400).

In Step 510 an appropriate primary page for comparison to a mirror page is determined. In one embodiment of the present invention, an atomic consistency element is examined and utilized to determine an appropriate primary page for comparison to a mirror page. If the atomic consistency element is zero, then the primary page associated with key[M] is the appropriate primary page for comparison. If the atomic consistency element is not zero the primary page associated with key[0] is appropriate, where the atomic consistency element refers to key[M]. In one embodiment of the present invention, if the atomic consistency element is non zero, then that nonzero atomic consistency element value is utilized as an ordinal value to determine which key[M] (e.g., "KEY[3]") is being referenced. The non zero atomic consistency element value is also utilized to determine which page the initial key[0] is on (e.g., "KEY[0]"). The genlock of the primary page including the initial key (e.g., KEY[0]) is used to override the genlock value of the primary page that includes a key[M]. This permits a change state to occur across the pages that include a key[M] and thereby performs an atomic change.

In step 520, the page with data to be utilized as valid data when recovering from a process or system crash is resolved by comparing the genlock of a mirror page to the genlock of an appropriate primary page

determined in step 510. If the genlock of both the appropriate primary page and the mirror page are unlocked, and the counter values are equal, then the data on both pages is consistent. If the genlock of the appropriate primary page is unlocked and the mirror page is locked, and the write counter value of the primary page is less than the write counter value of the mirror page, then the appropriate primary page comprises valid data. If the genlocks of the appropriate primary page and the mirror page are locked and the write counter value of the primary page and the write counter value of the mirror page are equal, then the mirror page comprises valid data. If the genlock of the appropriate primary page is unlocked and the mirror page is locked, and the write counter value of the primary page is less than the counter value of the mirror page, then the appropriate primary page comprises valid data. If the genlock of the appropriate primary page is unlocked and the mirror page is locked, and the write counter value of the primary page is greater than the counter value of the mirror page then the mirror page comprises valid data. An invalid state exists if the genlock of the appropriate primary page is locked and the mirror page is unlocked, and the data on the mirror page is considered valid and consistent.

In step 530 the consistency is restored by copying the invalid (inconsistent) page with the valid (consistent) page. To restore consistency, genlocks on the mirror and primary pages must be acquired. In one embodiment of the present invention, acquiring a genlock on a locked

genlock increases it by 2. Genlocks on the primary and mirror pages are then released. At the end of this process the associated primary and mirror genlocks will be equal.

Implementing data consistency maintenance method 400 and data consistency recovery method 500 together facilitates access to valid data and assurance that the data on a disk (e.g., a MDBM) is accessed in a consistent manner. Thus, when a process performs a read of data, the returned data is valid even if there is a system or software crash during an update of the database. The method also allows for a process that is performing a write operation in the database to determine what data is consistent (all writes completed successfully) and what data is not consistent. In one embodiment of the present invention data consistency maintenance method 400 and data consistency recovery method 500 allow an arbitrary process performing a write operation to either complete or abort the transaction that crashed.

In one embodiment of the present invention data is stored across multiple instances of a page based memory mapped, multiple anonymous reader, multiple writer database (e.g., a MDBM). In one exemplary implementation of the present invention, a MDBM allows for storage of multiple indexes onto the same data. In one embodiment of the present invention, storage of multiple indexes onto the same data is accomplished by storing the primary data in one database and each index stored in its own

MDBM database. In one exemplary implementation of the present invention, the storage of the data and the indexes may be of a nearly unlimited size. The value associated with a value that has a multi-key index is (implicitly) index 0. The values of the indexes are lists of keys for index 0.

Figure 6 is a block diagram illustrating one exemplary configuration of information stored in multiple index database system 600, one embodiment of the present invention. Multiple index database system 600 comprises index 610, index 650 and a third index (not shown). In one embodiment of the present invention, index 610, index 650 and a third index (not shown) are stored in separate databases. In one embodiment of the present invention, the indexes are stored in MDBM databases.

Index 610 comprises primary pages 611 through 613 and their associated mirror pages 614 through 616 respectively. A data value is broken into buckets identified by a reference to "KEY" and an appended identification (e.g., KEY00, KEY01, KEY02, KEY03 and KEY04). The first number in the appended identification is a reference to an index and the second number is a reference to the bucket number of that index. The buckets of data value referenced by "KEY" are stored over several pages including primary page 611 through 613 and copies stored in mirror pages 614 through 616. Each bucket starts at an ordinal element location (e.g., element location 0, 1, 2, etc.) within a page. Primary pages 611 through 613 and mirror pages 614 through 616

include write counter values 631 through 636, genlocks 641 through 646 and atomic consistency elements 621 through 626. In one embodiment of the present invention, the atomic consistency element values on the primary pages are set to a logical 0 value and do not change. The atomic consistency element values on the mirror page are set to correspond to an element location of a bucket (e.g., key[I,M]) being modified when the mirror page genlock is locked and returned to a logical zero when the mirror page genlock is unlocked.

Index 650 comprises primary pages 651 through 653 and their associated mirror pages 654 through 656 respectively. A data value is broken into buckets identified by a reference to KEY and an appended identification (e.g., KEY00, KEY01, KEY02, KEY03 and KEY04). The first number in the appended identification is a reference to an index and the second number is a reference to the bucket number of that index. The buckets of the data value referenced by "KEY" are stored over several pages including primary page 651 through 653 and copies storied in mirror pages 654 through 656. Each bucket starts at an ordinal element location (e.g., element location 0, 1, 2, etc.) within a page. Primary pages 651 through 653 and mirror pages 654 through 656 include write counter values 681 through 686, low order bits 691 through 696 and atomic consistency elements 671 through 676. In one embodiment of the present invention, the atomic consistency element values on the primary pages are set to a logical 0 value and do not change while the atomic

consistency element values on the mirror page are set to correspond to an element location of data being modified (e.g., key[I,M]) when the mirror page genlock is locked and returned to a logical zero when the mirror page genlock is unlocked.

Each bucket key [I,M] included in primary pages of a particular index has a primary object identifier (OID) associated with that index. In one exemplary implementation of the present invention, each primary object identifier is a unique value assigned to a bucket of value Key referenced in a particular index. Each bucket key [I,M] includes a list of secondary primary object identifiers that reference information in another index that should remain consistent with the value in bucket key [I,M]. For example, bucket KEY [01] of index 610 comprises particular data (e.g., a phone number) that is also included in bucket KEY [11] of index 650. Thus, the OID list included in Key 01 comprises a reference to OID20, the OID of index 650 associated with the particular information to the method maintains consistency for across both indexes.

In one embodiment of the present invention, a primary index comprises primary keys. In one exemplary implementation of the present invention, a primary key identifies a record in a table and has a value other than null. The primary keys (e.g., KEY [03]) of the primary index (e.g., index 610) are linked to a foreign key (e.g., KEY[13]) in another table (e.g., index 650).

The foreign key (e.g., KEY[13]) is an attribute that serves as the primary key (e.g., KEY[13]) of the other table (e.g., index 650) in the database.

Figure 7 is a flow chart of data consistency maintenance method 700, one embodiment of the present invention. Data consistency maintenance method 700 is a method for ensuring consistency of data stored across multiple instances of a page based memory mapped, multiple anonymous reader, multiple writer database (e.g., a MDBM). In one embodiment of the present invention, data consistency maintenance method 700 includes genlock and consistency methods for a single page and bucket chained data over multiple pages discussed above. In one embodiment of the present invention, data consistency maintenance method 700 is implemented on a computer system (e.g., data consistency maintenance and recovery computer system 100).

In data consistency method 700 a writer updates the mirror pages then updates the primary pages of databases for supplied indexes, then returns the databases to standard state. In one embodiment of the present invention, consistency state changes happen in an atomic manner. A small value is stored in conjunction with the MDBM genlock called the atomic consistency element. The nomenclature key[I] refers to the index I. The nomenclature key[I, M] refers to key (index) number I and bucket number M of that key.

In step 710, for each I in 0 through J, where J is the number of indexes associated with a value (e.g., referenced by KEY), the genlocks of previously unlocked mirror pages comprising information associated the value of indexed data are accessed and locked while updates to the information are performed and updated pages are synced to disk. In one embodiment of the present invention, step 710 is performed in two parts comprising step 711 and 713.

In step 711, for each M in 0 through N, where N is the number of buckets that the value associated with key[I] has been split into, the writing process acquires the genlock of the mirror page for key[I, M] (as described above), if the page was not previously locked. When modifying the genlock the atomic consistency element for this page is set to the element location for key[I, M] if M is greater than 0. Each page is synced to disk if the writer acquired the genlock.

In step 712, for each M in 0 through N, where N is the number of buckets that the value associated with key[I] has been split into, the writing process updates the mirror page and syncs the page to disk. If M equals 0 and I is greater 0, then the atomic consistency element is set to refer to key[I, M] and the update ensures that the first value (referring to index 0) listed is that of key[0]. In one embodiment of the present invention, an optimization

deferring the syncs on a per database basis until after all the pages have been updated allows coalescing the syncs of multiple buckets on the same page.

In step 720, for each I in 0 through J, where J is the number of indexes associated with a value, the genlocks of previously unlocked primary pages comprising information associated the value are accessed and locked while updates to the information are performed and updated primary pages are synced to disk. In one embodiment of the present invention step 720 is performed in two parts comprising step 721 and 723.

In step 721, for each M in 0 through N, where N is the number of buckets that the value associated with key[I] has been split into, the writing process acquires the genlock of the primary page for key[I, M] if the page was not previously locked. Each page is synced to disk if the writer acquired the genlock.

In step 723, for each M in 0 through N, where N is the number of buckets that the value associated with key[I] has been split into, the writing process updates the primary page and syncs the page to disk. In one embodiment of the present invention, an optimization deferring the syncs on a per database basis until after all the pages have been updated allows coalescing the syncs of multiple buckets on the same page.

In step 730, for each I in 0 through J, where J is the number of indexes associated with a value, the writing process releases the genlocks on the primary pages and the mirror pages and syncs them to disk. In one embodiment of the present invention, step 630 is performed in two steps, 731 and 733.

In step 731, for each M in 0 through N, where N is the number of buckets that the value associated with key[I] has been split into, the writing process releases the genlock on the primary page and syncs the page to disk.

In step 733, for each M in 0 through N, where N is the number of buckets that the value associated with a key has been split into, the writing process releases the genlock and sets the atomic consistency element to 0 on the mirror page and syncs the page to disk.

Data consistency method 700 guarantees that data that in memory is valid and the data on the disk is accessed in a consistent manner. Thus, when a process performs a read of data, the returned data is valid even if there is a system or software crash during an update of the database. The method also allows for a process that is performing a write in the database to determine what data is consistent (all writes completed successfully) and what data is not consistent. It also allows any arbitrary writer to either complete or abort the transaction that crashed.

Figure 8 is a flow chart of data consistency recovery method 800, one embodiment of the present invention. Data consistency recovery method 800 is a method for recovering valid bucket chained data across multiple pages of a page based memory mapped, multiple anonymous reader, multiple writer database (e.g., a MDBM). In one embodiment of the present invention, data is stored in multiple instances of MDBM files and not corrupted by process or system crashes that occur during a write process. In one embodiment of the present invention, data consistency recovery method 800 facilitates either roll back or completion of the changes to the pages in multiple MDBM databases in an atomic manner. Whether the data is rolled back after an aborted transaction depends upon point in the modification operation where the failure occurred. In one embodiment of present invention, data consistency recovery method 800 compares the genlocks of a mirror page with the appropriate primary page genlock to restore consistency. In one embodiment of the present invention, data consistency recovery method 800 is implemented in a computer system (e.g., data consistency maintenance and recovery computer system 100) utilizing a data consistency maintenance method (e.g., data consistency maintenance method 200).

In step 810 an appropriate index and primary page for comparison to a mirror page is determined. If the atomic consistency element is zero, then the primary page key[M] is the appropriate primary page for comparison. If M is

greater than 0 the primary page associated with key[I, 0] is appropriate, where the atomic consistency element refers to key[I, M]. Otherwise, the genlock is composed of primary page genlock key[I, M] logically "or-ed" with the low order bit of the primary genlock of key[0,0] of the first value in the value of key[I,0].

In one embodiment of the present invention, each page includes a second constructed genlock. A second constructed genlock is zero if the atomic consistency element is zero. A second constructed genlock is constructed if the atomic consistency element of a key in a given index I is non zero. That nonzero atomic consistency element value is utilized as an ordinal value to determine which key[I,M] (e.g., "KEY[13]") is being referenced and utilize that key[I,M] (e.g., "KEY[13]") to determine which page the initial key[I,0] is on (e.g., "KEY[I,0]"). The genlock of the primary page including the initial key (e.g., KEY[I,0]) is used to override the genlock value of the primary page that includes key[I,M]. This permits a change state to occur across the pages that include key[I,M] and thereby performs an atomic change.

In step 820 the page with data to be utilized as valid data when recovering from a crash is resolved by comparing the genlock of a mirror page to the genlock of an appropriate primary page determined in step 810. If the genlock of both the appropriate primary page and the mirror page are unlocked and the counter values are equal then the data on both pages are

consistent. If the genlock of the appropriate primary page is unlocked and the mirror page is locked and the write counter value of the primary page is less than the counter value of the mirror page then the appropriate primary page comprises valid data. If the genlocks of the appropriate primary page and the mirror page are locked and the write counter value of the primary page and the write counter value of the mirror page are equal then the mirror page comprises valid data. In the genlock of the appropriate primary page is unlocked and the mirror page is locked and the write counter value of the primary page is less than the counter value of the mirror page then the appropriate primary page comprises valid data. In the genlock of the appropriate primary page is unlocked and the mirror page is locked and the write counter value of the primary page is greater than the counter value of the mirror page then the mirror page comprises valid data. An invalid state exists if the genlock of the appropriate primary page is locked and the mirror page is unlocked. If an invalid state exists the data on the mirror page is considered valid and consistent.

In step 830 the consistency is restored by copying the invalid (inconsistent) page with the valid (consistent) page. To restore consistency, genlocks on the mirror and primary pages must be acquired. In one embodiment of the present invention, acquiring a genlock on a locked genlock increases it by 2. Genlocks on the primary and mirror pages are then

released. At the end of this process the associated primary and mirror genlocks will be equal.

A data maintenance and recovery system and process of the present invention comprises a variety of implementations in which consistency is maintained between two segments of data. In one exemplary implementation of the present invention, a data maintenance and recovery system and method is utilized to maintain and recover data consistency throughout distributed resources. In one embodiment, the present invention is utilized to maintain and recover data consistency in network communications. In one embodiment of the present invention, syncs are performed to a backing store or common point.

Thus, the present invention is a system and method that facilitates data consistency maintenance including during a write operation. The present invention also facilitates recovery from a system or process crash with valid data. A data consistency maintenance system and method of the present invention is adaptable for implementations maintaining data consistency throughout data on a single page, in buckets distributed over multiple pages and multiple data base instances of data referenced by multiple indexes.

The foregoing descriptions of specific embodiments of the present invention have been presented for purposes of illustration and description.

They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and obviously many modifications and variations are possible in light of the above teaching. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the Claims appended hereto and their equivalents.
